Homework 6: Q2

Name: Waiwai Kim, waiwaiki

1 Part (a): Algorithm Idea

Refer to Week 9 recitation notes. The algorithm idea is to place a tower at every 100 miles increment.

2 Part (a): Runtime Analysis

Here we are asked to proved that $d_{max} \leq 99n$. Week 9 recitation notes indicate that this implies while loop runs $99n/100 \leq n$ times, where n is the number of houses on the road. Note that if $d(a, b) = 100$ or the distance between two consecutive houses a, b is equal to 100 miles, this violates the assumption given to the problem. Thus, the maximum distance between two houses is 99 miles. If n houses exist on the road, the maximum distance of nth house or the last house from the first house is $99n$ miles. Because the inside of while loops increments by 100, this implies that the while loops runs $99n/100$ times.

3 Part (b): Algorithm Idea

The algorithm idea is to keep track of the maximum bandwidth or the bandwidth of the most recently installed tower. We will call this current bandwidth. If the next house is within the current bandwidth, there is no need to install a tower at the current location. Otherwise, we will install a tower at the current location. This implies that the current house is inside the current bandwidth. We will repeat to the second to the last house.

4 Part (b): Algorithm Details

In line 6 of Figure 1, the algorithm sort the input or house_list by ascending order of distances from the first house. After the sorting, the first element in the house_list is the second house closest to the first house at location 0.

In line 9-10 of Algorithm 1, we construct the list of tuples that the algorithm will return. It's assumes that the first tower is installed on the first house. Note that tower_list[i] = (house number where a tower is installed, bandwidth of the tower). For example, tower_list[0] == (0, 100) means that the first tower is installed at house 0 and the tower has bandwidth up to 100.

In line 12, the algorithm iterates from the second house to second-to-last house. In line 13, the algorithm checks if the next house is inside the current bandwidth. If so, the algorithm does nothing and go to the next house. In line 15-17, if the next house is not inside the current bandwidth, the algorithm installs a tower at the current house location and increment the current bandwidth by 100 miles. This location is added to the tower_list.

In line 19, after the algorithm finishes iterating through the houses, the algorithm returns the tower_list.

Figure 1: Tower Installment - Q2 Part b

```python
import sys
import operator

def q2_part_b(house_list):

    house_list.sort(key=operator.itemgetter(1))
    print("after sorting: " , house_list)

    tower_list= [(0,100)]
    cur_band = 100

    for i in range(len(house_list)-1):
        if house_list[i+1][1] < cur_band:
            pass
        else:
            cur_band = house_list[i][1] + 100
            tower_list.append((house_list[i][0], cur_band))

    return tower_list
```

5 Part (b): Proof of Correctness Idea

We have to prove that the above algorithm indeed results in the most cost effective installment or the minimum number of towers while ensuring every single house on the road gets high-speed internet.

First, we prove that every house gets broadband access. We know that the second house is inside the bandwidth from the first tower because 1) second house is less than 100 miles away from the first house and 2) the tower has 100 miles broadband distance. We can assume that a ith house is not inside the broadband and show contradiction using the argument shown above.

Second, we prove that the algorithm is indeed optimal by returning the minimum number of towers installed. We can assume there is an optimal solution with one less tower installment than our algorithm's solution and show that this contradicts the assumption that is a correct solution.

6 Part (b): Proof Details

First, we prove that every house gets broadband access. Assume that ith house do not receive broadband access. This implies either ith house is more than 100 miles from the $i-1$th house, which contradicts the constraints given to us. Then this implies that the $i-1$th house or previous houses that are less than 100 miles away from ith house do not have any tower. This contradicts the algorithm because the algorithm installs a tower if the next house is out of the current bandwidth. Thus, we prove that the algorithm indeed returns an installment plan that ensure ubiquitous broadband access.

Second, we prove that the algorithm indeed results in the minimum number of tower installments. Let's assume that there is an optimal solution with one less tower installment than our algorithm indicates. This implies that we can get rid of the additional $tower_i$ from our solution. We know that our algorithm added $tower_i$ at $house_i$ because $house_{i+1}$ was out of bound from the previous $tower_{i-1}$. Thus deleting $tower_i$ will results in a situation where

$house_{i+1}$ is out of $tower_{i-1}$'s range. This contradicts the assumption that this is an optimal solution. Thus we know that there cannot be another solution that has a fewer number of towers than our algorithm's solution, and we have proved that our solution is indeed correct.

7 Part (b): Runtime Analysis

The runtime analysis is simple. The first operation of the algorithm is to sort the house list by the ascending order of distances from the first house. We know that a sorting algorithm on an array, such as mergesort, has $O(nlogn)$ times. Line 9 and 10 can be assumed to take $o(1)$. Next, the line 12 of 1 runs $O(n)$ times because the for loop iterates index from 0 to $n-1$. We can assume that the comparison in line 13, addition in line 16 and appending in line 17 can take $O(1)$. Thus, the runtime of our algorithm is $O(n+nlogn) = O(nlogn)$.